TagEmbedSVD: Leveraging Tag Embeddings for Cross-Domain Collaborative Filtering

M. Vijaikumar(✉), Shirish Shevade, and M. N. Murty

Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, India
{vijaikumar,shirish,mnm}@iisc.ac.in

Abstract. Cross-Domain Collaborative Filtering (CDCF) mitigates data sparsity and cold-start issues present in conventional recommendation systems by exploiting and transferring knowledge from related domains. Leveraging user-generated tags (e.g. ancient-literature, military-history) for bridging the related domains is becoming a popular way for enhancing personalized recommendations. However, existing tag based models bridge the domains based on common tags between domains and their co-occurrence frequencies. This results in capturing the syntax similarities between the tags and ignoring the semantic similarities between them. In this work, to address these, we propose TagEmbedSVD, a tag-based CDCF model to cross-domain setting. TagEmbedSVD makes use of the pre-trained word embeddings (`word2vec`) for tags to enhance personalized recommendations in the cross-domain setting. Empirical evaluation on two real-world datasets demonstrates that our proposed model performs better than the existing tag based CDCF models.

Keywords: Cross-Domain Collaborative Filtering · User-generated tags

1 Introduction

Recommendation systems play an important role in filtering out irrelevant information and suggest contents that interest users. Examples include book recommendation in LibraryThing, movie recommendation in Netflix. Collaborative Filtering (CF) [9] has been successful in predicting user needs to be based on the like-minded users' interests using their historical records. However, in practice, users rate a very few as compared to available items. Besides, new users and/or items are added at regular intervals. These two phenomena lead to sparsity and cold-start problems.

To deal with the data sparsity and cold-start problems, researchers have exploited Cross-Domain Collaborative Filtering (CDCF) [3,7,8,15], which leverages the knowledge extracted from related domains. For example, users who watch movies from adventure genre will most likely be interested in reading books written on adventure travel. However, in most cases, users and/or items across domains do not overlap. In such a scenario, exploiting user-generated tags [2,4,5,11,13–16] (e.g. tags like ancient-literature or military-history) for bridging the related domains is becoming a popular way for enhancing personalized recommendations. That is, though we do not know the exact mapping of the users and/or items across domains, we establish the connections with the following assumption: *the users who use similar tags are similar, and the items which are assigned with similar tags are similar*. Nevertheless, existing tag based CDCF models work based on common tags and its co-occurrence count alone to bridge the related domains [5,13,14]. Hence, these models capture the syntax similarities between the tags and ignore the semantic relationships between them. We illustrate these by the following toy example.

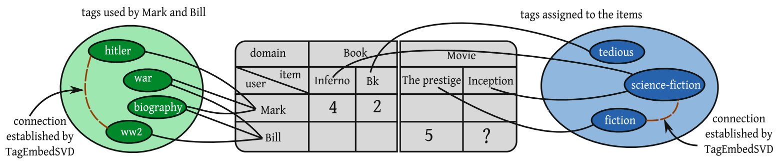
tags used by Mark and Bill
hitler
war
biography
ww2
connection established by TagEmbedSVD
domain
item
user
Book
Inferno
Bk
Movie
The prestige
Inception
Mark
Bill
4
2
5
?
tags assigned to the items
tedious
science-fiction
fiction
connection established by TagEmbedSVD

Fig. 1. Illustration of tag based CDCF

Let us assume that a cross-domain system has users (*Bill* and *Mark*), items (*The prestige* and *Inception* from Movie domain, and *Inferno* and *Bk* from book domain, respectively), ratings (in the scale of 1–5, where 1 being low and 5 being high) and their tag assignments as shown in Fig. 1.

Treating tags as tokens/lexicons, most of the existing models connect the users *Bill and Mark* by only the tags *biography and war* based on their direct usage. These models completely ignore the relationship between the tags *science-fiction* and *fiction* (as well as *ww2* and *hitler*). However, these tags provide a lot of knowledge about the similarities between the movies *The Prestige* and *Inception*, and also the movie *The Prestige* and the book *Inferno* since most users who are interested in movies or books related to *fiction* might also be interested in movies or books related to *science-fiction*. This can be well captured by word-vector embeddings such as `word2vec` [12], because the cosine similarity between the embedding vectors *fiction* and *science-fiction* is 0.61[1] which is considerably high. Besides the above facts on the similarity between items that can be established through tags, the similarity between the users *Mark* and *Bill* can also be strongly established with the help of semantic word embeddings, because the users are interested in similar topics *fiction* and *science-fiction* based on the items they rated. Accounting all of the above makes us predict the missing rating of the user *Bill* on the movie *Inception* more accurately.

[1] Here Google's pre-trained `word2vec` model trained on google news dataset is used for similarity calculation.

Contributions. We propose a novel extension of SVD++, TagEmbedSVD, to the cross-domain setting where there is no assumption that users and items overlap across domains. TagEmbedSVD leverages user-generated tags to bridge the related domains. That is, TagEmbedSVD employs `word2vec` – a word vector representation for finding the semantic relationship between the tags to bridge the domains and enhance the recommendation performance. We perform comprehensive experiments on two real-world datasets – LibraryThing and MovieLens, and show that our proposed model outperforms existing tag based CDCF models, particularly in sparse and cold-start settings. Our implementation is available at https://github.com/mvijaikumar/TagEmbedSVD.

2 Proposed Model

Problem Formulation. Suppose we have a set of ratings $[r^S_{uj}]_{m_S \times n_S}$ and $[r^T_{uj}]_{m_T \times n_T}$ from source and target domains. Here $r^S_{uj}, r^T_{uj} \in [a, b] \cup \{0\}$, and $a, b > 0$ denote the minimum and maximum ratings of user u on item j and 0 denotes unavailable ratings, respectively. Further, m_S, n_S, m_T and n_T represent the number of users and items from source and target domains respectively. In addition, we are given tags associated with every user u and item j denoted by sets $\mathcal{T}_u$ and $\mathcal{T}_j$ respectively. Here, the tag can be a word or a set of words or a phrase (for example, philosophy, mind-blowing, one time watchable).

Let U^S, U^T, I^S and I^T denote the sets of users and items from source and target domains respectively. Note that, $U = U^S \cup U^T$ and $I = I^U \cup I^T$, and $U^S \cap U^T = \varnothing$ and $I^S \cap I^T = \varnothing$. Let $\mathcal{I}_u$ and $\mathcal{U}_j$ be the set of items rated by user u and the set of users who rated item j respectively. Let $\Omega^S = \{(u, j) : r^S_{uj} > 0\}$ and $\Omega^T = \{(u, j) : r^T_{uj} > 0\}$ be the sets indicating the available ratings and $\Omega = \Omega^S \cup \Omega^T$. Our goal here is to predict the unavailable ratings for the users on items in the target domain with the help of available ratings and tag information from both domains. Formally, we want to predict ratings r^T_{uj}, $\forall (u, j) \notin \Omega^T$ in the target domain using r^S_{uj}, $\forall (u, j) \in \Omega^S, r^T_{uj}$, $\forall (u, j) \in \Omega^T$ and $\mathcal{T}_u, \mathcal{T}_j$, $\forall u \in U$, $\forall j \in I$. To avoid notational clutter, wherever the context is clear from Ω^S and Ω^T, we drop the superscripts S and T and combine the ratings from the source and target domains together.

2.1 TagEmbedSVD

In this section, we explain our proposed model – TagEmbedSVD, in detail. TagEmbedSVD is an extension of SVD++ [9]. The main objective here is to incorporate knowledge learned from tags into the SVD++ model for cross-domain settings. In this way, tags are able to not only provide additional knowledge to understand the system but also able to bridge the users and items across domains. Let $p_u \in \mathbb{R}^d$ be user (u) embedding and $q_j \in \mathbb{R}^d$ and $y_j \in \mathbb{R}^d$ be item (j) embeddings, $\mu \in \mathbb{R}, b_u \in \mathbb{R}$ and $b_j \in \mathbb{R}$ be the mean value of all the available ratings, user bias and item bias, respectively. Let $t_k \in \mathbb{R}^c$ be embedding vector associated with tag k, and c denote the embedding dimension. Here, we predict rating $\hat{r}_{uj}$ as follows:

$$\hat{r}_{uj} = \mu + b_u + b_j + (p_u + |\mathcal{I}_u|^{-\frac{1}{2}} \sum_{i \in \mathcal{I}_u} y_i)' q_j + \frac{\alpha}{|\mathcal{T}_u|} \sum_{k \in \mathcal{T}_u} w'_u E t_k + \frac{\beta}{|\mathcal{T}_j|} \sum_{k \in \mathcal{T}_j} x'_j F t_k, \tag{1}$$

where $w_u \in \mathbb{R}^d$ captures user u's preferences towards the tags and $x_j \in \mathbb{R}^d$ is item j's characteristics towards the tags. We obtain embeddings for tags from `word2vec` [12]. Thus, any two tags k and k' can be compared by its embeddings t_k and t'_k from the start of the training. Due to this fact, if two users share similar tag preferences, irrespective of their domain difference, we can obtain preference of users on tags from different domains. That is, p_u and q_j are learned from only the available ratings from the corresponding domains since users (items) from different domains do not share any item (user) in common. However, w_u and x_j are learned combinedly irrespective of its domain difference through the tag embeddings. Here, α and β control the influence of tags on predictions. In our model, to share a common embedding space and to have flexibility in choosing the dimension of t_k, we use projection matrices E and $F \in \mathbb{R}^{d \times c}$.

The number of occurrences of the tags associated with users and items plays an important role in characterising users and items. To adapt this knowledge we modify the definition $|\mathcal{T}_u|$ to be $\sum_{k \in \mathcal{T}_u} \eta_{uk}$, where η_{uk} denotes the frequency of the tag t_k associated with the user u. Similarly, we define $|\mathcal{T}_j|$ to be $\sum_{k \in \mathcal{T}_j} \eta_{jk}$. Therefore, Eq. (1) becomes:

$$\begin{aligned} \hat{r}_{uj} = \mu + b_u + b_j + (p_u + |\mathcal{I}_u|^{-\frac{1}{2}} \sum_{i \in \mathcal{I}_u} y_i)' q_j + \\ \frac{\alpha}{|\mathcal{T}_u|} \sum_{k \in \mathcal{T}_u} \eta_{uk} w'_u E t_k + \frac{\beta}{|\mathcal{T}_j|} \sum_{k \in \mathcal{T}_j} \eta_{jk} x'_j F t_k. \end{aligned} \tag{2}$$

Additionally, we use a weighted-regularization technique [6,9] to control over-fitting which arises due to the sparsity issue. Here, popular users and items are penalized less since more ratings are available, and users and items having less ratings are penalized more since only a few ratings are available for them. For instance, we regularize the user representation p_u by multiplying scalar $|\mathcal{I}_u|^{-\frac{1}{2}}$ instead of $|\mathcal{I}_u|$. Note that, the former penalizes less on users who rated more items as compared to the latter also the weighted-regularization does not drop any user or item. Let λ and λ_M be the positive hyperparameters to control over-fitting. We have the following optimization problem:

$$\min_{p_*,q_*,y_*,x_*,w_*,b_*} \mathcal{L} = \frac{1}{2} \sum_{(u,j)\in \Omega^S \cup \Omega^T} (\hat{r}_{uj} - r_{uj})^2 + \frac{\lambda}{2} (\sum_u |\mathcal{I}_u|^{-\frac{1}{2}} b_u^2 + \sum_u |\mathcal{U}_j|^{-\frac{1}{2}} b_j^2) + \frac{\lambda}{2} \sum_u |\mathcal{I}_u|^{-\frac{1}{2}} (||p_u||^2 + ||w_u||^2) + \frac{\lambda}{2} \sum_j |\mathcal{U}_j|^{-\frac{1}{2}} (||q_j||^2 + ||x_j||^2) + \frac{\lambda}{2} \sum_i |\mathcal{U}_i|^{-\frac{1}{2}} ||y_i||^2 + \frac{\lambda_M}{2} (||E||_F^2 + ||F||_F^2). \tag{3}$$

Complexity Analysis. Let a_I and a_T be the average number of rated items by users and average number of tags used by users (or tags assigned to items). Naive implementation of TagEmbedSVD takes $O(a_I|\Omega|d + a_T|\Omega|dc)$ to compute the objective value in Eq. (3) since we project tags into lower dimension using E and F. However, we can use hashtable to store and retrieve the sum of the embeddings of the tags corresponding to each user and item and this results in $O(a_I|\Omega|d + |\mathcal{T}^{uniq}|dc)$ time complexity, where $|\mathcal{T}^{uniq}|$ represents the number of unique tags in the system. Note that, in real time $|\mathcal{T}^{uniq}|dc << a_I|\Omega|d$. In addition, gradient computational effort of TagEmbedSVD requires $O(a_I|\Omega|d + d|\mathcal{T}^{uniq}|c + a_T|\Omega|dc)$ whereas SVD++ requires $O(a_I|\Omega|d)$. This is due to bottleneck in computing $\frac{\partial \mathcal{L}}{\partial E}, \frac{\partial \mathcal{L}}{\partial F}$. It can be reduced considerably by updating $\frac{\partial \mathcal{L}}{\partial E}, \frac{\partial \mathcal{L}}{\partial F}$ after some fixed number of intervals instead of every iteration. In our experiments, we observed that updating E and F after every ten iterations does not degrade the performance significantly.

Table 1. Dataset statistics

	#users	#items	#ratings	sparsity	#tag-assignments
MovieLens (ML)	5000	5000	2,68,092	98.93%	54,830
LibraryThing(LT)	5000	5000	1,05,171	99.58%	2,96,829

3 Experiments

Datasets and Evaluation Methodology. We used two publicly available datasets – MovieLens-10M[2] and LibraryThing[3] for cross-domain collaborative filtering setting. Statistics of the datasets are given in Table 1. For constructing training and validation split pairs, we follow the same procedure as used in [13,14]. From the target domain, we extract $K\%$ of the available ratings for the training set and the remaining $(100-K)\%$ ratings are used for either validation or test purpose. We extract six such pairs. The first pair is used for tuning the hyperparameters, that is, the corresponding left out $(100-K)\%$ ratings act as a

[2] https://grouplens.org/datasets/movielens/10m/.
[3] http://www.macle.nl/tud/LT/.

Table 2. Performance comparison for **setting 1**.

Setting 1	Metric	TagCDCF	GTagCDCF	TagGSVD++	TagEmbedSVD	Improvement (%)
ML (all)	MAE	0.6911	0.6905*	0.6926	**0.6524**	5.84
source (LT)	RMSE	0.8922	0.8915*	0.8920	**0.8439**	5.64
ML (cold-start users)	MAE	0.8652	0.7943*	0.7962	**0.7421**	7.03
source (LT)	RMSE	1.0786	1.0048*	1.0267	**0.9370**	7.24
ML (cold-start items)	MAE	0.6925	0.6895	0.6841*	**0.6526**	4.83
source (LT)	RMSE	0.8910	0.8901*	0.8914	**0.8434**	5.54
LT(all)	MAE	0.6658	0.6621	0.6543*	**0.6329**	3.38
source (ML)	RMSE	0.8618	0.8412*	0.8476	**0.8192**	2.69
LT (cold-start users)	MAE	0.8066	0.7503	0.7468*	**0.7141**	4.58
source (ML)	RMSE	1.0284	0.9652*	0.9928	**0.9321**	3.55
LT (cold-start items)	MAE	0.6697	0.6602	0.6556*	**0.6333**	3.52
source (ML)	RMSE	0.8646	0.8387*	0.8486	**0.8184**	2.48

Table 3. Performance comparison for **setting 2**.

Setting 2	Metric	TagCDCF	GTagCDCF	TagGSVD++	TagEmbedSVD	Improvement (%)
ML (all)	MAE	0.7117	0.7183	0.7095*	**0.6783**	4.60
source (LT)	RMSE	0.9146	0.9226	0.9135*	**0.8738**	4.54
ML (cold-start users)	MAE	0.8646	0.8009*	0.8282	**0.7767**	3.12
source (LT)	RMSE	1.0850	1.0241*	1.0839	**0.9949**	2.93
ML (cold-start items)	MAE	0.7121*	0.7163	0.7135	**0.6779**	5.04
source (LT)	RMSE	0.9150*	0.9197	0.9196	**0.8723**	4.90
LT(all)	MAE	0.7037	0.6828*	0.7053	**0.6556**	4.15
source (ML)	RMSE	0.9059	0.8635*	0.9027	**0.8411**	2.66
LT (cold-start users)	MAE	0.7971	0.7260*	0.7549	**0.6881**	5.49
source (ML)	RMSE	1.0051	0.9201*	0.9711	**0.8894**	3.45
LT (cold-start items)	MAE	0.7065	0.6830*	0.7017	**0.6554**	4.21
source (ML)	RMSE	0.9075	0.8632*	0.9009	**0.8411**	2.63

validation set. Once the hyperparameters values are obtained, we train the other five pairs with these values and obtain the test set performance. Here, in these five pairs, left out $(100 - K)\%$ ratings act as a test set. We report the average test error obtained from these five test sets as the final performance. During these extractions, we make sure that there exists at least one rating for all the users and items in the training set.

We conduct experiments under the following settings. In all the above settings, we add the source domain ratings (if multiple source domains are available we combine their ratings together) as a part of the training set.

1. **Setting 1**: We set $K = 80$.
2. **Setting 1, cold-start users (items)**: This is same as **setting 1**, but, results corresponding to only cold-start users (items) are reported.
3. **Setting 2**: Here, we set $K = 40$ to introduce more sparsity in the target domain part of the training set.
4. **Setting 2, cold-start users (items)**: This is same as **setting 2**, but, results corresponding to only cold-start users (items) are reported.

Further, **all** in Tables 2 and 3 indicates that all the users and items are included in test set, where as, **cold-start users** (**cold-start items**) indicates only users who rated less than five items (items which received ratings from less than five users) are included in test set. Similar definitions are followed in [6]. In Tables 2 and 3 bold-faced value indicates best performance, and 'Improvement' indicates the relative improvements that TagEmbedSVD achieves against the best performance among the comparison models highlighted by symbol *.

Comparison of Models. We compare our model with the following tag based CDCF models.

1. **TagCDCF** [13] extends matrix factorization and leverages tag information to improve the performance by understanding the similarities between users and items with the help of common tags.
2. **GTagCDCF** [14] connects the source and target domain by common tags. It additionally takes the frequency of the tag usage into account.
3. **TagGSVD++** [5] is an extension to SVD++ model. It uses tag information in place of implicit feedback in SVD++ to obtain user and item representations.

Since it has been demonstrated in [13,14] that the tag based CDCF models perform better than single domain models we do not include them for comparison.

Metrics. We employ two well-known metrics, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for performance comparisons [6,9,10].

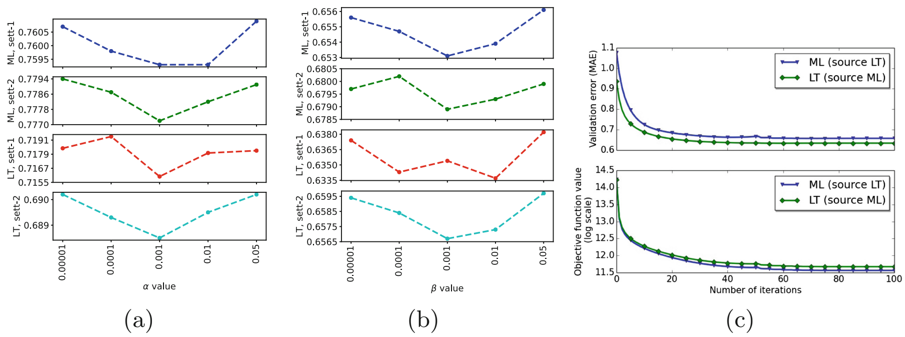

Fig. 2. Impact of parameter (a) α and (b) β on cold-start users and items respectively. Here y-axis indicates MAE value.

Parameter Setting. We tune our hyperparameters using random hyperparameter search [1] and validation set with 100 trials for each model. For TagEmbedSVD, we tune λ from [0.01, 2], α from [0.00001, 0.05], β from [0.00001, 0.05] and latent dimension (d) from $\{5, 10, 20, 30, 40\}$. Ranges for comparison models were selected from the respective papers [5,13,14].

3.1 Results

Performance: Tables 2 and 3 compare the performance of TagEmbedSVD with the other tag-based CDCF models. We conduct a paired t-test and all the improvements are statistically significant for $p < 0.01$. The main findings from Tables 2 and 3 are summarized as follows:

1. TagEmbedSVD performs better than the other models by up to 5.84% (setting 1) and up to 4.60% (setting 2) when all the users and items are used. Similarly, it gives improvements up to 7.24% (setting 1) and 5.49% (setting 2) for cold-start users, and 5.54% (setting 1) and 5.04% (setting 2) for cold-start items respectively. This demonstrates the significance of using distributed representations for tags to improve the performance within and across domains.
2. One of the main reasons for TagEmbedSVD's better performance than that of TagCDCF and GTagCDCF is the utilization of all the available tags instead of just common tags. Further, despite both being extensions to SVD++, we gain improvement in TagEmbedSVD over TagGSVD++. The reason is that the former treats tags as tokens, hence *fiction* and *science-fiction* (or *hitler* and *ww2*) are two different tags. Whereas, the latter utilizes the pre-trained distributed representations for the tags, hence, they are very close to each other in the embedding space.

Impact of Parameters α and β: We investigate the effect of parameters α and β in Eq. (2), that control the influence of tag information to TagEmbedSVD. Note that, letting α and β to zero, it results in the SVD++ model. For cold-start users in datasets ML and LT, we fixed the other hyperparameter values and varied α in both setting 1, and setting 2. As we increase the value of α, the performance of the model improves as shown in Fig. 2(a). If α is set to the higher value, the performance decreases. It is because the information comes from tags dominates the rating values. We get similar behavior from the parameter β in cold-start item setting. This is illustrated in Fig. 2(b). Further, objective function value with respect to number of iterations are given in Fig. 2(c).

4 Conclusion

In this paper, we propose a simple and easy to train tag based cross-domain collaborative filtering model – TagEmbedSVD for leveraging tag information to cross-domain recommendations. TagEmbedSVD differs from the other models by employing distributed representations for tags to bridge the source and target domains. In this way, any two tags can be compared. Our experimental results show that our model performs better than other tag-based models in various sparse and cold-start settings. Although we use a single source and single target domain, TagEmbedSVD is general and any number of domains can be used without further modifications in the model.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(Feb), 281–305 (2012)
2. Bharti, R., Gupta, D.: Recommending top N movies using content-based filtering and collaborative filtering with hadoop and hive framework. In: Kalita, J., Balas, V.E., Borah, S., Pradhan, R. (eds.) Recent Developments in Machine Learning and Data Analytics. AISC, vol. 740, pp. 109–118. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-1280-9_10
3. Cantador, I., Fernández-Tobías, I., Berkovsky, S., Cremonesi, P.: Cross-domain recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 919–959. Springer, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_27
4. Fang, Z., Gao, S., Li, B., Li, J., Liao, J.: Cross-domain recommendation via tag matrix transfer. In: ICDMW, pp. 1235–1240. IEEE (2015)
5. Fernández-Tobías, I., Cantador, I.: Exploiting social tags in matrix factorization models for cross-domain collaborative filtering. In: CBRecSys, pp. 34–41 (2014)
6. Guo, G., Zhang, J., Yorke-Smith, N.: TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: AAAI (2015)
7. He, M., Zhang, J., Yang, P., Yao, K.: Robust transfer learning for cross-domain collaborative filtering using multiple rating patterns approximation. In: WSDM, pp. 225–233 (2018)
8. Khan, M.M., Ibrahim, R., Ghani, I.: Cross domain recommender systems: a systematic literature review. ACM Comput. Surv. (CSUR) **50**(3), 36 (2017)
9. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: SIGKDD, pp. 426–434. ACM (2008)
10. Li, B., Yang, Q., Xue, X.: Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In: IJCAI, vol. 9, pp. 2052–2057 (2009)
11. Lops, P., Jannach, D., Musto, C., Bogers, T., Koolen, M.: Trends in content-based recommendation. User Model. User-Adapted Interact. **29**, 239–249 (2019)
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
13. Shi, Y., Larson, M., Hanjalic, A.: Tags as bridges between domains: improving recommendation with tag-induced cross-domain collaborative filtering. In: Konstan, J.A., Conejo, R., Marzo, J.L., Oliver, N. (eds.) UMAP 2011. LNCS, vol. 6787, pp. 305–316. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22362-4_26
14. Shi, Y., Larson, M., Hanjalic, A.: Exploiting social tags for cross-domain collaborative filtering. arXiv preprint arXiv:1302.4888 (2013)
15. Shi, Y., Larson, M., Hanjalic, A.: Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput. Surv. (CSUR) **47**(1), 3 (2014)
16. Wang, W., Chen, Z., Liu, J., Qi, Q., Zhao, Z.: User-based collaborative filtering on cross domain by tag transfer learning. In: Workshop on Cross Domain Knowledge Discovery in Web and Social Network Mining, pp. 10–17. ACM (2012)